UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/726,902	12/03/2003	Mitchell Alsup	5500-88700	4177

53806 7590 02/06/2007
MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL (AMD)
P.O. BOX 398
AUSTIN, TX 78767-0398

EXAMINER	
FENNEMA, ROBERT E	
ART UNIT	**PAPER NUMBER**
2183	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	02/06/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

2

Office Action Summary	Application No.	Applicant(s)	
	10/726,902	ALSUP ET AL.	
	Examiner	Art Unit	
	Robert E. Fennema	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *20 November 2006*.
2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.
3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-35* is/are pending in the application.
4a) Of the above claim(s) _____ is/are withdrawn from consideration.
5)☐ Claim(s) _____ is/are allowed.
6)☒ Claim(s) *1-35* is/are rejected.
7)☐ Claim(s) _____ is/are objected to.
8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.
10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
a)☐ All b)☐ Some * c)☐ None of:
1.☐ Certified copies of the priority documents have been received.
2.☐ Certified copies of the priority documents have been received in Application No. _____.
3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date *1/5/07;10/4/06*.
4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-35 have been considered. Claims 1, 7, 14, 20, and 32 have been amended as per Applicant's request.

Information Disclosure Statement

2. The reference in the IDS submitted 10/4/2006 by Mendelson et al. has not been considered, as the document number is incomplete. Examiner was not able to find the document, and thus has not considered the reference.

3. The foreign reference in the IDS submitted 1/5/2007 has not been considered as only an abstract was submitted with the IDS, and as the Examiner has not been able to read the reference as a whole or the portions quoted in the foreign examination report, and thus has not considered it.

Claim Objections

4. Claims 1 and 14 are objected to for the use of the term "a predicted target address". In Claim 1, this term appears in the amended sections, after another "a predicted target address" has been established. It is not clear if Applicant is referring to a different predicted target address, or the previously established one, however, Examiner is interpreting the "a" as "the" for the purposes of examination, in line with the first amended limitation of Claim 14, and Claim 32. Applicant is required to confirm that the language in these Claims are correct, and if not, to correct them accordingly. If the

language is correct, Examiner strongly suggests adding adjectives such as "first", "second", or "next", to help clearly identify and make clear that the predicted target addresses are intended to be different, as it is currently unclear what predicted target addresses are being used and responded to by the trace cache.

5. Claims 27 and 35 are objected to, as the final limitation does not appear to limit the claim. Each claim states that a new trace is started if the received operation is at a branch label, but that the new trace is delayed if a previous trace is a duplicate, until the received operation is a branch label. Therefore, the claims appear to state that a trace is only started on a branch label, and if there is a duplicate, a trace is still only started on a branch label, which does not change the fact that the trace is only started on a branch label. Examiner believes Applicant is attempting to indicate that a duplicate trace continues to trace until a branch label is encountered, irregardless of when the duplicate is discovered, and has interpreted the claim as such based on Applicants arguments and specification, however, correction to both claims are required to clarify these issues.

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

7. Claims 1-2,11-15, and 24-26 are rejected under 35 U.S.C. 102(b) as being anticipated by Rotenberg et al. (herein Rotenberg).

8. As per Claim 1, Rotenberg teaches: A microprocessor (Abstract), comprising:

an instruction cache configured to store instructions (Section 2.1, first paragraph);

a branch prediction unit (Section 2.1, first paragraph);

a trace cache configured to store a plurality of traces of instructions (Section 2.2, second paragraph); and

a prefetch unit coupled to the instruction cache, the branch prediction unit, and the trace cache (Figure 4, and Section 2.2. The Instruction latch appears to fill the role of a prefetch unit);

wherein the prefetch unit is configured to fetch instructions from the instruction cache until the branch prediction unit outputs a predicted target address (Section 2.2, where it is said that "on a trace cache miss, fetching proceeds normally from the instruction cache); and

wherein the prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting a predicted target address (inherent in Rotenberg);

wherein the prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs a predicted target address (inherent in Rotenberg); and wherein if the prefetch unit identifies a match for the predicted target address in the trace cache, the prefetch unit is configured to fetch one or more of the plurality of traces from the trace cache (Section 2.2, where it is said that "on a trace cache hit, an entire trace of instructions is fed into the instruction latch, bypassing the instruction cache).

9. As per Claim 2, Rotenberg teaches: The microprocessor of claim 1, wherein the branch prediction unit is configured to output the predicted target address in response to a prediction that a branch will be taken (Section 2.1, which discloses "They (the BTB banks) serve the role of detecting branches in the instructions currently being fetched and providing their target addresses, in time for the next fetch cycle. A few paragraphs further down, it is stated that this happens "if there is a predicted taken branch").

10. As per Claim 11, Rotenberg teaches: The microprocessor of claim 1, wherein each of the plurality of traces comprises partially-decoded instructions (it is inherent that a trace comprises a partially-decoded instruction, otherwise the necessary control information as show in section 2.2 would not be available).

11. As per Claim 12, Rotenberg teaches: The microprocessor of claim 1, wherein each of the plurality of traces is associated with a tag comprising the address of an earliest instruction, in program order, stored within that trace (Section 2.2, where the tag identifies the starting address of the trace).

12. As per Claim 13, Rotenberg teaches: The microprocessor of claim 1, wherein each of the plurality of traces is associated with a flow control field comprising a label for an instruction to which control will pass for each branch operation comprised in that trace (Section 2.2, the "trace target address" and "trace fall-through address" are both labels which describe where control will flow based on each branch operation, based on the prediction (which is part of another label, "branch flags")).

13. As per Claim 14, Rotenberg teaches: A computer system, comprising:

a system memory (inherent if the system has an instruction cache); and

a microprocessor coupled to the system memory (also inherent in Rotenbergs invention), comprising:

an instruction cache configured to store instructions (Section 2.1, first paragraph);

a branch prediction unit (Section 2.1, first paragraph);

a trace cache configured to store a plurality of traces of instructions (Section 2.2); and

a prefetch unit coupled to the instruction cache, the branch prediction unit, and the trace cache (Figure 4, and Section 2.2. The Instruction latch appears to fill the role of a prefetch unit);

wherein the prefetch unit is configured to fetch instructions from the instruction cache until the branch prediction unit outputs a predicted target address (Section 2.2, where it is said that "on a trace cache miss, fetching proceeds normally from the instruction cache);

wherein the prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting the predicted target address (inherent in Rotenberg);

wherein the prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs a predicted target address (inherent in Rotenberg); and

wherein if the prefetch unit identifies a match for the predicted target address in the trace cache, the prefetch unit is configured to fetch one or more of the plurality of traces from the trace cache (Section 2.2, where it is said that "on a trace cache hit, an entire trace of instructions is fed into the instruction latch, bypassing the instruction cache).

14. As per Claim 15, Rotenberg teaches: The computer system of claim 14, wherein the branch prediction unit is configured to output the predicted target address in response to a prediction that a branch will be taken (Section 2.1, which discloses "They (the BTB banks) serve the role of detecting branches in the instructions currently being fetched and providing their target addresses, in time for the next fetch cycle. A few paragraphs further down, it is stated that this happens "if there is a predicted taken branch").

15. As per Claim 24, Rotenberg teaches: The computer system of claim 14, wherein each of the plurality of traces comprises partially-decoded instructions (it is inherent that a trace comprises a partially-decoded instruction, otherwise the necessary control information as show in section 2.2 would not be available).

16. As per Claim 25, Rotenberg teaches: The computer system of claim 14, wherein each of the plurality of traces is associated with a tag comprising the address of an earliest instruction, in program order, stored within that trace (Section 2.2, where the tag identifies the starting address of the trace).

17. As per Claim 26, Rotenberg teaches: The computer system of claim 14, wherein each of the plurality of traces is associated with a flow control field comprising a label for an instruction to which control will pass for each branch operation comprised in that trace (Section 2.2, the "trace target address" and "trace fall-through address" are both labels which describe where control will flow based on each branch operation, based on the prediction (which is part of another label, "branch flags")).

Claim Rejections - 35 USC § 103

18. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

19. Claims 3, 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, in view of Patterson et al. (herein Patterson).

20. As per Claim 3, Rotenberg teaches the microprocessor of claim 1, but fails to teach: wherein the branch prediction unit is configured to output the predicted target address in response to detection of a branch misprediction. However, Patterson teaches that in order to reduce the penalty for a misprediction, you can fetch both the taken and not taken instructions, and put them in the BTB, which would then be able to immediately output the correct path on a misprediction without having to fetch it (Pages 276-277). While it would increase the cost of the system, the advantage is a smaller misprediction penalty, which may worth the cost, depending on the needs of the system. Therefore, one of ordinary skill in the art at the time the invention was made would have stored both the taken and not taken branch paths in the BTB in order to reduce the misprediction penalty, and thus increasing performance.

21. As per Claim 16, Rotenberg teaches the computer system of claim 14, but fails to teach: wherein the branch prediction unit is configured to output the predicted target address in response to detection of a branch misprediction. However, Patterson teaches that in order to reduce the penalty for a misprediction, you can fetch both the taken and not taken instructions, and put them in the BTB, which would then be able to immediately output the correct path on a misprediction without having to fetch it (Pages 276-277). While it would increase the cost of the system, the advantage is a smaller

misprediction penalty, which may worth the cost, depending on the needs of the system. Therefore, one of ordinary skill in the art at the time the invention was made would have stored both the taken and not taken branch paths in the BTB in order to reduce the misprediction penalty, and thus increasing performance.

22. Claims 4, 10, 17, 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, in view of Braught, further in view of Xia.

23. As per Claim 4, Rotenberg teaches: The microprocessor of claim 1, further comprising a trace generator (it is necessary for Rotenbergs invention to have a trace generator in order to create traces), but fails to teach:

wherein the trace generator is configured to begin a trace with an instruction corresponding to a label boundary.

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not specifically teach that an instruction must be a branch to access the cache (despite requiring branch prediction bits). However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches.

However, this combination still fails to teach starting a trace on a label boundary. Braught teaches that in Assembly Language, most branches operate on labels, which the machine can only interpret when converted to an address (Page 3). In Braughts example on page 3, all branches are labels, making every branch a label boundary. When combined with Rotenbergs invention, this would mean that a trace would be generated with an instruction corresponding to a label boundary, as all the branches would go to labels, and even an address may be interpreted as a label. Given this knowledge, it would have been obvious to one of ordinary skill in the art that Rotenbergs invention begins traces when it encounters an instruction with a label boundary, as it only begins on branch instructions, which branch to labels.

24. As per Claim 10, Rotenberg teaches: The microprocessor of claim 4, wherein the trace generator is configured to generate traces in response to instructions being decoded (Section 2.2, the trace can not be completed (written into the cache) until the trace target addresses are calculated, which requires the instructions to be decoded).

25. As per Claim 17, Rotenberg teaches: The computer system of claim 14, further comprising a trace generator (it is necessary for Rotenbergs invention to have a trace generator in order to create traces), but fails to teach:

wherein the trace generator is configured to begin a trace with an instruction corresponding to a label boundary.

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not specifically teach that an instruction must be a branch to access the cache (despite requiring branch prediction bits). However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches.

However, this combination still fails to teach starting a trace on a label boundary. Braught teaches that in Assembly Language, most branches operate on labels, which the machine can only interpret when converted to an address (Page 3). In Braughts example on page 3, all branches are labels, making every branch a label boundary. When combined with Rotenbergs invention, this would mean that a trace would be generated with an instruction corresponding to a label boundary, as all the branches would go to labels, and even an address may be interpreted as a label. Given this knowledge, it would have been obvious to one of ordinary skill in the art that Rotenbergs invention begins traces when it encounters an instruction with a label boundary, as it only begins on branch instructions, which branch to labels.

26. As per Claim 23, Rotenberg teaches: The computer system of claim 17, wherein the trace generator is configured to generate traces in response to instructions being decoded (Section 2.2, the trace can not be completed (written into the cache) until the trace target addresses are calculated, which requires the instructions to be decoded).

27. Claims 5-8 and 18-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, Xia and Braught, further in view of Lange et al. (USPN 3,896,419, herein Lange).

28. As per Claim 5, Rotenberg, Xia and Braught teach the microprocessor of claim 4, but fail to teach:

wherein the trace generator is configured to check the trace cache for a duplicate copy of the trace that the trace generator is constructing.

Rotenberg teaches a standard trace cache, with the potential for some duplication, when considering the alternative embodiment of path associativity disclosed in Section 2.3. And while Rotenberg does not discuss the issue of duplication, he does teach of disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and that this situation needs to be avoided, otherwise features which increase cost or decrease performance must be implemented, and further teaches the disadvantages of a useless trace displacing a useful trace (Section 2.3, judicious trace selection). Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces, and also a system which requires action when a miss occurs while servicing a miss. Lange teaches a system in which a cache is checked for a value while accessing memory, and if the identical value is found in the cache, the fetch to memory is aborted, freeing the memory to be used by a different operation (preventing a blocking load, Column 9, Lines 60-65) This is analogous to Rotenberg in the sense that Lange provides a motivation to look for duplicates in a cache to prevent a large delay and allow another operation to continue (analogous to finding a solution to what to do when encountering a miss while servicing a miss) by implementing a solution of searching for the duplicate, and not performing a long-latency operation on the duplicate value since it is not required (if the previous trace appears to be a duplicate of what is in the trace, there is no need to continue tracing, and the duplicate value can be discarded as taught by Lange (there is no need to perform an action on a duplicate value), preventing the line-fill buffer from blocking new traces, a long-latency event).

In an alternative viewpoint disclosed above, Rotenberg also teaches why it is disadvantageous to displace a useful trace with a useless trace (under the interpretation that a duplicate trace being stored in the trace cache is useless, as the original can always be accessed in its place), thus, with the potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments, Rotenberg indicates in his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used, providing a motivation to prevent these duplicate traces from entering the trace cache from the line fill buffer, further providing a motivation to find some method to handle these cases, which Lange provides, by simply discarding the duplicate value.

Given these advantages in both of these situations, one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to Rotenberg, in order to solve either or both of the problems of getting a miss while servicing a miss, and preventing eviction of a useful trace by a duplicate trace.

29. As per Claim 6, Lange teaches: The microprocessor of claim 5, wherein if the trace generator identifies a duplicate copy of the trace, the trace generator is configured to discard the trace under construction (Column 5, Lines 5-10).

30. As per Claim 7, Rotenberg teaches: The microprocessor of claim 5, wherein in response to the trace generator identifying an entry corresponding to a duplicate copy of the trace, the trace generator is configured to check the trace cache for an entry corresponding to a next trace to be generated (Section 2.2. The trade cache is checked every time there is a potential new trace, so when one trace is found and discarded, the next potential new trace will cause the trace cache to be checked again).

31. As per Claim 8, Lange teaches: The microprocessor of claim 7, wherein if the trace generator identifies a trace entry corresponding to the next trace to be generated,

the trace generator is configured to discard the trace under construction (Column 5, Lines 5-10).

32. As per Claim 18, Rotenberg, Xia and Braught teach the computer system of claim 17, but fail to teach:

wherein the trace generator is configured to check the trace cache for a duplicate copy of the trace that the trace generator is constructing.

Rotenberg teaches a standard trace cache, with the potential for some duplication, when considering the alternative embodiment of path associativity disclosed in Section 2.3. And while Rotenberg does not discuss the issue of duplication, he does teach of disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and that this situation needs to be avoided, otherwise features which increase cost or decrease performance must be implemented, and further teaches the disadvantages of a useless trace displacing a useful trace (Section 2.3, judicious trace selection). Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces, and also a system which requires action when a miss occurs while servicing a miss. Lange teaches a system in which a cache is checked for a value while accessing memory, and if the identical value is found in the cache, the fetch to memory is aborted, freeing the memory to be used by a different operation (preventing a blocking load, Column 9, Lines 60-65) This is analogous to Rotenberg in the sense that Lange provides a motivation to look for duplicates in a cache to prevent a large delay and allow another operation to continue (analogous to finding a solution to what to do when encountering a miss while servicing a miss) by implementing a solution of searching for the duplicate, and not performing a long-latency operation on the duplicate value since it is not required (if the previous trace appears to be a duplicate of what is in the trace, there is no need to continue tracing, and the duplicate value can be discarded as taught by Lange (there is no need to perform an action on a duplicate value), preventing the line-fill buffer from blocking new traces, a long-latency event).

In an alternative viewpoint disclosed above, Rotenberg also teaches why it is disadvantageous to displace a useful trace with a useless trace (under the interpretation that a duplicate trace being stored in the trace cache is useless, as the original can always be accessed in its place), thus, with the potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments, Rotenberg indicates in his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used, providing a motivation to prevent these duplicate traces from entering the trace cache from the line fill buffer, further providing a motivation to find some method to handle these cases, which Lange provides, by simply discarding the duplicate value.

Given these advantages in both of these situations, one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to Rotenberg, in order to solve either or both of the problems of getting a miss while servicing a miss, and preventing eviction of a useful trace by a duplicate trace.

33. As per Claim 19, Lange teaches: The computer system of claim 18, wherein if the trace generator identifies a duplicate copy of the trace, the trace generator is configured to discard the trace under construction (Column 5, Lines 5-10).

34. As per Claim 20, Rotenberg teaches: The computer system of claim 18, wherein in response to the trace generator identifying an entry corresponding to a duplicate copy of the trace, the trace generator is configured to check the trace cache for an entry corresponding to a next trace to be generated (Section 2.2. The trade cache is checked every time there is a potential new trace, so when one trace is found and discarded, the next potential new trace will cause the trace cache to be checked again).

35. As per Claim 21, Lange teaches: The computer system of claim 20, wherein if the trace generator identifies a trace entry corresponding to the next trace to be generated, the trace generator is configured to discard the trace under construction (Column 5, Lines 5-10).

36. Claims 9 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, Xia, and Braught, in view of Akkary et al. (USPN 6,247,121, herein Akkary).

37. As per Claim 9, Rotenberg teaches the microprocessor of claim 4, but fails to teach:

wherein the trace generator is configured to generate traces in response to instructions being retired.

While Rotenberg teaches that the instructions need to be decoded (Section 2.2) before the trace can be generated, it is not taught that they need to be retired beforehand (although it is suggested in Section 2.3, fill issues). Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (Column 3, Lines 40-44). Rotenberg discusses in Section 2.3 that some traces are committed but never used, thus evicting a useful trace. By ensuring that the trace is correct, the odds that it will be useful is increased, as an incorrect (not actually executed) trace should probably not need to be used, as branches tend to behave the same way (Section 1.1). Therefore, one of ordinary skill in the art at the time the invention was made would have waiting until an instruction was retired before considering adding it to the trace, in order to ensure accuracy of the trace and to prevent displacement of trace more likely to be reused.

38. As per Claim 22, Rotenberg teaches the computer system of claim 17, but fails to teach:

wherein the trace generator is configured to generate traces in response to instructions being retired.

While Rotenberg teaches that the instructions need to be decoded (Section 2.2) before the trace can be generated, it is not taught that they need to be retired beforehand (although it is suggested in Section 2.3, fill issues). Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (Column 3, Lines 40-44). Rotenberg discusses in Section 2.3 that some traces are committed but never used, thus evicting a useful trace. By ensuring that the trace is correct, the odds that it will be useful is increased, as an incorrect (not actually executed) trace should probably not need to be used, as branches tend to behave the same way (Section 1.1). Therefore, one of ordinary skill in the art at the time the invention was made would have waiting until an instruction was retired before considering adding it to the trace, in order to ensure accuracy of the trace and to prevent displacement of trace more likely to be reused.

39. Claims 27-31 and 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, Xia, Braught, and Lange, further in view of Akkary.

40. As per Claim 27, Rotenberg teaches: A method, comprising:

starting construction of a new trace (Section 2.2), but fails to teach:

starting a new trace if the received instruction is associated with a branch label;

receiving a retired instruction; and

if a previous trace under construction duplicates a trace in a trace cache, delaying construction of the new trace until the received instruction corresponds to a branch label.

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not specifically teach that an instruction must be a branch to access the cache (despite requiring branch prediction bits). However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches.

However, this combination still fails to teach starting a trace on a label boundary. Braught teaches that in Assembly Language, most branches operate on labels, which the machine can only interpret when converted to an address (Page 3). In Braughts example on page 3, all branches are labels, making every branch a label boundary. When combined with Rotenbergs invention, this would mean that a trace would be generated with an instruction corresponding to a label boundary, as all the branches would go to labels, and even an address may be interpreted as a label. Given this knowledge, it would have been obvious to one of ordinary skill in the art that Rotenbergs invention begins traces when it encounters an instruction with a label boundary, as it only begins on branch instructions, which branch to labels.

While Rotenberg teaches that the instructions need to be decoded (Section 2.2) before the trace can be generated, it is not taught that they need to be retired beforehand (although it is suggested in Section 2.3, fill issues). Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (Column 3, Lines 40-44). Rotenberg discusses in Section 2.3 that some traces are committed but never used, thus evicting a useful trace. By ensuring that the trace is correct, the odds that it will be useful is increased, as an incorrect (not actually executed) trace should probably not need to be used, as branches tend to behave the same way (Section 1.1). Therefore, one of ordinary skill in the art at the time the invention was made would have waiting until an instruction was retired before considering adding it to the trace, in order to ensure accuracy of the trace and to prevent displacement of trace more likely to be reused.

Rotenberg teaches a standard trace cache, with the potential for some duplication, when considering the alternative embodiment of path associativity disclosed in Section 2.3. And while Rotenberg does not discuss the issue of duplication, he does teach of disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and that this situation needs to be avoided, otherwise features which increase cost or decrease performance must be implemented, and further teaches the disadvantages of a useless trace displacing a useful trace (Section 2.3, judicious trace selection). Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces, and also a system which requires action when a miss occurs while servicing a miss. Lange teaches a system in which a cache is checked for a value while accessing memory, and if the identical value is found in the cache, the fetch to memory is aborted, freeing the memory to be used by a different operation (preventing a blocking load, Column 9, Lines 60-65) This is analogous to Rotenberg in the sense that Lange provides a motivation to look for duplicates in a cache to prevent a large delay and allow another operation to continue (analogous to finding a solution to what to do when encountering a miss while servicing a miss) by implementing a solution of searching for the duplicate, and not performing a long-latency operation on the duplicate value since it is not required (if the previous trace appears to be a duplicate of what is in the trace, there is no need to continue tracing, and the duplicate value can be discarded as taught by Lange (there is no need to perform an action on a duplicate value), preventing the line-fill buffer from blocking new traces, a long-latency event).

In an alternative viewpoint disclosed above, Rotenberg also teaches why it is disadvantageous to displace a useful trace with a useless trace (under the interpretation that a duplicate trace being stored in the trace cache is useless, as the original can always be accessed in its place), thus, with the potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments, Rotenberg indicates in his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used, providing a motivation to prevent these duplicate traces from entering the trace cache from the line fill buffer, further providing a motivation to find some method to handle these cases, which Lange provides, by simply discarding the duplicate value.

Given these advantages in both of these situations, one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to Rotenberg, in order to solve either or both of the problems of getting a miss while servicing a miss, and preventing eviction of a useful trace by a duplicate trace.

The fact that traces can only begin on branch labels means that the next trace cannot be started until an instruction represents a branch label, irregardless of when the duplication is found and dealt with.

41. As per claim 28, Rotenberg teaches: The method of claim 27, further comprising continuing construction of an incomplete trace already in process (Section 2.2).

42. As per Claim 29, Lange teaches: The method of claim 27, further comprising searching the trace cache for duplicate entries (Column 5, Lines 5-10).

43. As per Claim 30, Rotenberg teaches: The method of claim 29, further comprising creating a new entry in the trace cache if no duplicate entry is identified (Section 2.2).

44. As per Claim 31, Lange teaches: The method of claim 29, further comprising discarding a trace if a duplicate entry is identified (Column 5, Lines 5-10).

45. As per Claim 35, Rotenberg teaches: A microprocessor comprising:

means for starting a new trace (Section 2.2), but fails to teach:

starting a new trace if the received operation is a first operation at a branch label;

means for receiving a retired operation;

means for delaying starting a new trace if a previous trace under construction duplicates a trace in trace cache, until the received operation corresponds to a branch label.

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not specifically teach that an instruction must be a branch to access the cache (despite requiring branch prediction bits). However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches.

However, this combination still fails to teach starting a trace on a label boundary. Braught teaches that in Assembly Language, most branches operate on labels, which the machine can only interpret when converted to an address (Page 3). In Braughts example on page 3, all branches are labels, making every branch a label boundary. When combined with Rotenbergs invention, this would mean that a trace would be generated with an instruction corresponding to a label boundary, as all the branches would go to labels, and even an address may be interpreted as a label. Given this knowledge, it would have been obvious to one of ordinary skill in the art that Rotenbergs invention begins traces when it encounters an instruction with a label boundary, as it only begins on branch instructions, which branch to labels.

While Rotenberg teaches that the instructions need to be decoded (Section 2.2) before the trace can be generated, it is not taught that they need to be retired beforehand (although it is suggested in Section 2.3, fill issues). Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (Column 3, Lines 40-44). Rotenberg discusses in Section 2.3 that some traces are committed but never used, thus evicting a useful trace. By ensuring that the trace is correct, the odds that it will be useful is increased, as an incorrect (not actually executed) trace should probably not need to be used, as branches tend to behave the same way (Section 1.1). Therefore, one of ordinary skill in the art at the time the invention was made would have waiting until an instruction was retired before considering adding it to the trace, in order to ensure accuracy of the trace and to prevent displacement of trace more likely to be reused.

Rotenberg teaches a standard trace cache, with the potential for some duplication, when considering the alternative embodiment of path associativity disclosed in Section 2.3. And while Rotenberg does not discuss the issue of duplication, he does teach of disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and that this situation needs to be avoided, otherwise features which increase cost or decrease performance must be implemented, and further teaches the disadvantages of a useless trace displacing a useful trace (Section 2.3, judicious trace selection). Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces, and also a system which requires action when a miss occurs while servicing a miss. Lange teaches a system in which a cache is checked for a value while accessing memory, and if the identical value is found in the cache, the fetch to memory is aborted, freeing the memory to be used by a different operation (preventing a blocking load, Column 9, Lines 60-65) This is analogous to Rotenberg in the sense that Lange provides a motivation to look for duplicates in a cache to prevent a large delay and allow another operation to continue (analogous to finding a solution to what to do when encountering a miss while servicing a miss) by implementing a solution of searching for the duplicate, and not performing a long-latency operation on the duplicate value since it is not required (if the previous trace appears to be a duplicate of what is in the trace, there is no need to continue tracing, and the duplicate value can be discarded as taught by Lange (there is no need to perform an action on a duplicate value), preventing the line-fill buffer from blocking new traces, a long-latency event).

In an alternative viewpoint disclosed above, Rotenberg also teaches why it is disadvantageous to displace a useful trace with a useless trace (under the interpretation that a duplicate trace being stored in the trace cache is useless, as the original can always be accessed in its place), thus, with the potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments, Rotenberg indicates in his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used, providing a motivation to prevent these duplicate traces from entering the trace cache from the line fill buffer, further providing a motivation to find some method to handle these cases, which Lange provides, by simply discarding the duplicate value.

Given these advantages in both of these situations, one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to Rotenberg, in order to solve either or both of the problems of getting a miss while servicing a miss, and preventing eviction of a useful trace by a duplicate trace.

The fact that traces can only begin on branch labels means that the next trace cannot be started until an instruction represents a branch label, irregardless of when the duplication is found and dealt with.

46. Claims 32-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, in view of Xia.

47. As per Claim 32, Rotenberg teaches: A method, comprising:

fetching instructions from an instruction cache (Section 2.1, paragraphs 1-3);

continuing to fetch instructions from the instruction cache until a branch target address is generated (Section 2.2, where it is said that "on a trace cache miss, fetching proceeds normally from the instruction cache);

in response to a branch target address being generated, searching trace cache for an entry corresponding to the branch target address (Section 2.2, third paragraph), but fails to teach:

without searching a trace cache;

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not teach not searching the trace cache until a branch target address is generated, and in fact teaches that the trace cache is searched on every instruction. However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage of using less cache space, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches. An effect of only having traces start on branches is that it is then inherent that a non-branch instruction can never be the start of a trace, therefore there would be no reason to search the trace cache for a non-branch instruction, which has further potential advantages such as power saving and potential critical path reduction, as one of ordinary skill in the art would recognize.

48. As per Claim 33, Rotenberg teaches: The method of claim 32, further comprising continuing to fetch instructions from the instruction cache if no entry is identified in the trace cache corresponding to the branch target address (Section 2.2, where it is said that "on a trace cache miss, fetching proceeds normally from the instruction cache).

As per Claim 34, Rotenberg teaches: The method of claim 32, further comprising fetching one or more traces from the trace cache if an entry is identified in the trace cache corresponding to the branch target address (Section 2.2, where it is said that "on a trace cache hit, an entire trace of instructions is fed into the instruction latch, bypassing the instruction cache).

Response to Arguments

49. Regarding Claim 1, and for the same reasons, Claim 14, the amendments to the claim are an inherency to the invention as disclosed by Rotenberg. The claims in question essentially disclose the limitations of "checking the trace cache for a match for a predicted target address in response to the branch prediction unit outputting a predicted target address, and where the trace cache is not checked for a match until the prediction unit outputs a predicted address". It is inherent that you cannot check for a value if you do not know what the value is. The trace cache as disclosed by Rotenberg cannot search for the predicted target address in the cache if it does not know what the

predicted target address is, and in fact, no cache or any other type of hardware can find something when it doesn't know what it is looking for, thus, the value must have been output from the branch prediction unit prior to the checking.

50. Regarding the arguments concerning Claims 27 and 35, Applicant has argued two points, firstly, that Akkary does not teach using a retired instruction for any particular purpose, and that the motivation of "incorrect" traces is not valid. Secondly, Applicant essentially argues that duplicate values cannot exist in Rotenberg, and thus do not need to be considered or checked for, and also that the performance problems solved by Lange do not apply to Rotenberg.

Regarding the first issue about retired instructions, Akkary teaches that instructions are retired when they are correctly executed, additionally, this is something that is well known in the art to one of ordinary skill in the art, as retirement is a term given to instructions that are committing to the register file, and will not be found to be incorrect later, thus writing to the register file can never be undone and cause memory consistency issues, Akkary was used to demonstrate this definition in practice, and further evidence of this definition is found at Column 9, Lines 59-61, and Column 10, Lines 2-6 (where deallocation of the trace buffers are a step of instruction retirement, however, Akkary uses trace buffers for a slightly different purpose than Rotenberg, thus the deallocation). Given that a retired instruction is an instruction that has been guaranteed to have been executed, the issue of an "incorrect" trace is addressed. The idea behind Examiners statement of an "incorrect" trace is a trace which does not represent the flow of the instruction stream, that is, it does not represent the series of branches that occurred in correct execution of the program, but rather, an alternate series of instructions which did not occur. While this data may be valuable in a trace cache, as it will provide the correct path in the event that the path will be a correct execution in the future, it does not represent what did happen, and Rotenbergs motivation for creating the trace cache was to exploit code reuse, both points being given in Section 1.1, that an instruction which was used recently will be used again in the future, and that branches tend to be biased in one directions, and it is likely that certain paths will be followed frequently. Clearly, if a trace is storing an outcome which was incorrect, while potentially useful in the future, it will not exploit either of those two cases, because it will not be likely to occur again. It is possible the incorrect trace would be used again, but it is not likely, and given that all memory and space is limited, one would recognize that it would be favorable to store traces likely to be used as opposed to not be used (Also see section 2.3, Point 6, where it can be seen that storing infrequently used traces could displace useful traces). Therefore, the incorrect trace represents a trace, that while possible, did not actually occur, and the advantage for tracing only retired instructions vastly increases the likelihood of reuse, based on the two factors disclosed earlier, and providing a motivation to combine Akkary with Rotenberg. Furthermore, Rotenberg himself suggests the possibility of using retired instructions instead of speculative results in Section 2.3, Point 5, in the last sentence. However, the language in the claim rejection has been amended to make this point more clear.

Regarding Applicant's second arguments regarding Lange and duplicate values, Examiner asserts that duplicate values can exist in Rotenberg's trace cache implementation, evidence of this is found in Section 2.3, point 2, regarding multiple paths. In this alternative embodiment, multiple traces from the same starting point are traced, as they may diverge later in the path, thus two traces from a single starting address can be stored in the trace cache. However, it is possible the paths will not diverge, and that the trace under construction will be the same as traces already in the cache, it cannot be known until it is complete, thus the potential for duplicates exist because of the need to trace everything to account for the possibility of divergence. Given this disclosure, Examiner has reinterpreted Rotenberg and Lange, and has put a new grounds of rejection for the claims in question, with a clearer motivation to combine and has further expanded on the problems and solutions provided by the references.

These same arguments also apply to the claims which claim similar subject matter, for example 5, 18, and 29.

However, Examiner would like to point out that in both Claims 27 and 35, on closer examination, the final limitation does not appear to limit the claim. Each claim states that a new trace is started if the received operation is at a branch label, but that the new trace is delayed if a previous trace is a duplicate, until the received operation is a branch label. Therefore, the claims appear to state that a trace is only started on a branch label, and if there is a duplicate, a trace is still only started on a branch label, which does not change the fact that the trace is only started on a branch label. Examiner believes Applicant is attempting to indicate that a duplicate trace continues to trace until a branch label is encountered, irregardless of when the duplicate is discovered, and has interpreted the claim as such, however, correction to both claims are required to clarify these issues.

51. Regarding Claim 6, Applicant has argued that the act of blocking the data retrieval from memory does not teach discarding a duplicate trace. Examiner disagrees, the limitations are analogous. Lange teaches aborting the read from memory of the data, thus preventing the retrieval of the duplicate data, and therefore it is discarded.

52. Regarding Claim 7, the trace cache is checked in response to identifying a duplicate trace, as the trace cache is always checked for a next trace entry, it is in response because it will always occur, unless Applicant is trying to claim that the trace cache is not checked for a next entry unless a duplicate was found, which would appear to contradict both Claim 1 and the specification.

53. Regarding Claim 8, the combination of references disclosed in the previous claims teach finding the next trace to be generated, and why the trace would be discarded upon discovery of the duplicate value.

54. Regarding Claims 9, 10, 22 and 23. Examiner asserts that something being a necessary condition is the same as initiating a response. If a trace requires that an instruction be retired in order to generate a trace, it can clearly not enter the trace until it

has been retired, once it has been retired, it can enter/start the trace, making it responsive on the act of it retiring. The same logic applies to decoding.

55. Applicant's arguments concerning Claim 32 are moot due to the new grounds of rejection necessitated by the amendment.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Robert E. Fennema whose telephone number is (571) 272-2748. The examiner can normally be reached on Monday-Friday, 8:45-6:15.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Robert E Fennema
Examiner
Art Unit 2183

RF

[illegible]

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100